# IMAGE SUPER-RESOLUTION RECONSTRUCTION

Contents



## 1. Introduction

The goal is to reconstruct an image with a higher resolution than the original resolution of the used sensor. For this aim, we suppose that $n$ images are recorded, where each images differs from small translations. This problem can be formulated as a *linear inverse problem* and has real application in real life. For example, smartphone can embed several sensors to provide high resolution images.

## 2. Generalities about linear inverse problems

A linear inverse problem is defined *a contrario* by its *direct model*. Let an original signal $\mathbf{x} \in \mathbb{R}^N$. This signal is measured thanks to a linear operator $\mathbf{A} \in \mathbb{R}^{M \times N}$ providing the observation $\mathbf{y} \in \mathbb{R}^M$ which can be corrupted by an additive noise $\mathbf{n} \in \mathbb{R}^M$. The direct model is the following:

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{n}$$

An inverse problem is said to be "well posed" in the Hadamard's sense if:

(1) There exists a solution
(2) The solution is unique
(3) The solution's behavior changes continuously with the initial conditions

If one of these conditions is violated, the problem is said to be ill-posed.

Two special cases can be considered:

(1) $M \geq N$: the problem is said to be over-determined. In that case, the first condition if usually violated.
(2) $M \leq N$: the problem is said to be under-determined. In that case, the second condition is always violated. The third condition can also be violated.

The goal of an inverse problem is to get a nice estimation of $\mathbf{x}$ given $\mathbf{y}$ and $\mathbf{A}$, that is to "invert" the operator $\mathbf{A}$.

2.1. **Inverse problem and pseudo-inversion.** Suppose that the noise is negligible, that is the direct problem reads:

$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

The most direct method in order to get an estimation of $\mathbf{x}$, is to use the Moore-Penrose pseudo-inverse of $\mathbf{A}$.

(1) If the problem is over-determined ($M \geq N$), $\mathbf{A}$ admits a left pseudo-inverse given by

$$\mathbf{A}^\dagger = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$$

which is equivalent to the least-square solution:

$$\begin{aligned}\mathbf{x}^{est} &= \underset{\mathbf{x}\in\mathbb{R}^N}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{A}\mathbf{x}\| \\ &= (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{y} \\ &= \mathbf{A}^\dagger\mathbf{y}\end{aligned}$$

The difficulties come when the matrix $\mathbf{A}^T\mathbf{A}$ is ill-conditionned, that is the inversion is numerically unstable.

(2) If the problem is under-determined ($M \leq N$), and $\operatorname{rank}(\mathbf{A}) = M$, then $\mathbf{A}$ admits an infinity of right inverse. The Moore-Penrose pseudo-inverse is given by

$$\mathbf{A}^\dagger = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}$$

leading to the minimum energy solution:

$$\begin{aligned}\mathbf{x}^{est} &= \underset{\mathbf{x}\in\mathbb{R}^N}{\operatorname{argmin}} \|\mathbf{x}\|^2 \text{ s.t.} \mathbf{y} = \mathbf{A}\mathbf{x} \\ &= \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{y} \\ &= \mathbf{A}^\dagger\mathbf{y}\end{aligned}$$

2.2. **The cost functional framework.** The pseudo inverse rarely leads to the best estimate. One possible, general, framework is to construct a cost functional for which the minimizer leads to an estimation of the original signal $\mathbf{x}$. The functional can be constructed by using:

(1) A loss, or data fidelity term, $\mathcal{L}(\mathbf{x})$. This loss links the observation $\mathbf{y}$ and the original signal $\mathbf{x}$ through the system $\mathbf{A}$. It aims to model the additive noise $\mathbf{n}$. The most common choice, which is well adapted to gaussian white noise, is to chose a $\ell_2$ loss:

$$\mathcal{L}(\mathbf{y}) = \frac{1}{2} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|^2$$

(2) A regularizer, or prior, $\mathcal{R}(\mathbf{x})$. This prior aims to model the knowledge of the original signal $\mathbf{x}$. Several choices are possible:
- $\ell_2$ norm, leading to the ridge regression a.k.a the minimum norm solution:
$$\mathcal{R}(\mathbf{x}) = \frac{1}{2} \|\mathbf{x}\|_2^2$$
- $\ell_2$ norm on the derivative, to obtain smooth solution:
$$\mathcal{R}(\mathbf{x}) = \frac{1}{2} \|\nabla\mathbf{x}\|_2^2$$
where $\nabla$ is simply the difference operator.
- $\ell_1$ norm, leading to the Lasso, if the sought solution is sparse:
$$\mathcal{R}(\mathbf{x}) = \|\mathbf{x}\|_1$$
- $\ell_1$ norm on the derivative, to obtain piecewise constant solution (well adapted for images):
$$\mathcal{R}(\mathbf{x}) = \|\nabla\mathbf{x}\|_1$$
- A combination of previous possibilities
- etc.

(3) A hyperparameter $\lambda \in \mathbb{R}_+$, to balance between the loss and the regularizer. When the additive noise is a white gaussian noise with a known variance $\sigma_0^2$ and the loss is $\ell_2$, one can tune this parameter in order to get a residual with a variance close to $\sigma_0^2$.

Then, the estimation of $x$ is given by

$$\mathbf{x}^{est} = \underset{\mathbf{x}\in\mathbb{R}^N}{\operatorname{argmin}} \mathcal{L}(\mathbf{x}) + \lambda\mathcal{R}(\mathbf{x})$$

For example, with an $\ell_2$ loss and a smooth derivative prior, the estimation leads to:

$$\mathbf{x}^{est} = \underset{\mathbf{x}\in\mathbb{R}^N}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{y} - \mathbf{A}\mathbf{x}\|^2 + \lambda\frac{1}{2} \|\nabla\mathbf{x}\|_2^2$$

2.3. **Use of a dictionary.** Sometimes, it can be convenient to introduce a *dictionary*, such as wavelets or time-frequency, in order to obtain *sparse* representation of the signal and then improving the prior. Let $\mathbf{\Phi} \in \mathbb{R}^{N\times K}$ be such a dictionary[1]. Then the direct model becomes

$$\mathbf{y} = \mathbf{A}\mathbf{x} = \mathbf{A}\mathbf{\Phi}\boldsymbol{\alpha}$$

[1] $\mathbf{\Phi}$ can be chosen in $\mathbb{C}^{N\times K}$, but we stick to real representation for the sake of simplicity, and without loss of generality.

where $\boldsymbol{\alpha}$ are the *synthesis* coefficients of $\mathbf{x}$. If the dictionary is well chosen, one can suppose that this synthesis coefficients are sparse, and then use the functionnal

$$\boldsymbol{\alpha}^{est} = \underset{\mathbf{x}\in\mathbb{R}^N}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{y} - \mathbf{A}\boldsymbol{\Phi}\boldsymbol{\alpha}\|^2 + \lambda \|\boldsymbol{\alpha}\|_1$$

and recover the estimate by

$$\mathbf{x}^{est} = \boldsymbol{\Phi}\boldsymbol{\alpha}^{est}$$

2.4. **Algorithm.** Now that we are able to construct a cost functional in order to get an estimate, one must chose an efficient algorithm to minimize it. A general approach is to use proximal descent algorithm, which have several advantages :

- it is simple to implement
- it is fast
- it is easy to adapt

Proximal descent algorithm can tackle functional of the form:

$$f(\mathbf{x}) + g(\mathbf{x})$$

where

- $f$ is a convex Lipschitz differentiable function (such as $\ell_2$ norm)
- $g$ is convex, possibly non smooth function (such as $\ell_1$ norm)

It relies on the so called proximity operator of $g$:

$$\text{prox}_g(\mathbf{z}) = \underset{\mathbf{x}\in\mathbb{R}^N}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{z} - \mathbf{x}\|_2^2 + g(\mathbf{x})$$

For example, if $g(\mathbf{x}) = \lambda\|\mathbf{x}\|_1$, then the proximity operator is given coordinatewise by

$$\forall k,\ x_k = z_k \left(1 - \frac{\lambda}{|z_k|}\right)^+$$

with $(x)^+ = \max(0, x)$.

Proximal descent algorithm then reads

**Algorithm 1:** Proximal descent

Initialization: $x \in \mathbb{R}^N$, $t = 0$

**repeat**

(1) $\mathbf{z}^{(t)} = \mathbf{x}^{(t)} - \nabla_f(\mathbf{x}^{(t)})$

(2) $\mathbf{x}^{(t+1)} = \text{prox}_g(\mathbf{z}^{(t)})$

(3) $\mathbf{z}^{(t+1)} = \mathbf{x}^{(t+1)} + \frac{t}{t+4}\left(\mathbf{x}^{(t+1)} - \mathbf{x}^{(t)}\right)$

**until** *convergence*;

In particular, if $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2^2$, one has $\nabla_f(\mathbf{x}) = -\mathbf{A}^T(\mathbf{y} - \mathbf{A}\mathbf{x})$.

**Exercice 1.** *Identify the appropriate $f$ and $g$, and adapt the proximal descent algorithm in the following cases:*

*(1)* $\mathcal{L}(\boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{y} - \mathbf{A}\boldsymbol{\Phi}\boldsymbol{\alpha}\|$ *and* $\mathcal{R}(\boldsymbol{\alpha}) = \|\boldsymbol{\alpha}\|_1$

*(2)* $\mathcal{L}(\mathbf{x}) = \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|$ *and* $\mathcal{R}(\mathbf{x}) = \|\mathbf{x}\|_2^2$

*(3)* $\mathcal{L}(\mathbf{x}) = \frac{1}{2}\|\mathbf{y} - \mathbf{A}\mathbf{x}\|$ *and* $\mathcal{R}(\mathbf{x}) = \|\nabla\mathbf{x}\|_2^2$

## 3. Two particular "sub" inverse problems

3.1. **Inpainting.** Inpainting consists of automatically fill some holes in images. The direct model can then be written as

$$\mathbf{y} = \mathbf{M}(\mathbf{x} + \mathbf{n})$$

where

- $\mathbf{x} \in \mathbb{R}^N$ is the original image
- $\mathbf{y} \in \mathbb{R}^N$ is the corrupted image
- $\mathbf{n} \in \mathbb{R}^N$ is some additive white Gaussian noise
- $\mathbf{M} \in \mathbb{R}^{N \times N}$ is a *binary mask*: $M[k] = 0$ if the pixel $k$ is hidden, and $M[k] = 1$ if the pixel $k$ is observed.

We consider here a special case of inpainting, where the binary mask $\mathbf{M}$ is generated randomly thanks to a Bernoulli distribution with parameter $p$. That is $M[k] = 1$ with probability $p$ and $M[k] = 0$ with probability $1 - p$.

**Exercice 2.**

*(1) Generate a random binary mask* $\mathbf{M}$ *for a chosen parameter* $p$.
*(2) Generate the direct problem for various Gaussian white noise*
*(3) Using an* $\ell_2$ *loss, try various regularizer (and* $\lambda$ *hyperparameter) to estimate the original image* $\mathbf{x}$

3.2. **Deconvolution.** Deconvolution consists of restoring a blurred image, given the kernel of the blur. The direct model can be written as

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{n}$$

where

- $\mathbf{x} \in \mathbb{R}^N$ is the original image
- $\mathbf{y} \in \mathbb{R}^M$ is the blurred image, with $M < N$
- $\mathbf{n} \in \mathbb{R}^M$ is some additive white Gaussian noise
- $\mathbf{C} \in \mathbb{R}^{M \times N}$ correspond to the blurring linear operator (the convolution operation). If we measure $M$ data in the signal $y$ from a convolution with an impulse response of size $P$, we then have used $N = M + P - 1$ data from the original signal $x$.

The size $N$ of the original data $\mathbf{x}$ is then bigger from the measured signal. Consequently, the convolution operator admits several left inverse and the deconvolution problem is an under-determined inverse problem.

Using matlab and python, the direct operation $\mathbf{C}\mathbf{x}$ can be implement on images using `conv2` or `scipy.signal.convolve2d` with the option `'valid'`. The adjoint operator (that is $\mathbf{C}^T\mathbf{y}$) can be implemented using the option `'full'` applied to the *flipped* (up–down, and left–right in the 2D case) impulse response.

**Exercice 3.**

*(1) Simulate the direct problem using the PSF data with various Gaussian white noise*
*(2) Using an* $\ell_2$ *loss, try various regularizer (and* $\lambda$ *hyperparameter) to estimate the original image* $\mathbf{x}$

## 4. Super-resolution: direct model

4.1. **General direct model.** We describe here the direct model of the super-resolution inverse problem, i.e. how several "low" resolution images are obtained from one high resolution image. This direct model can be summarized on Fig. 1

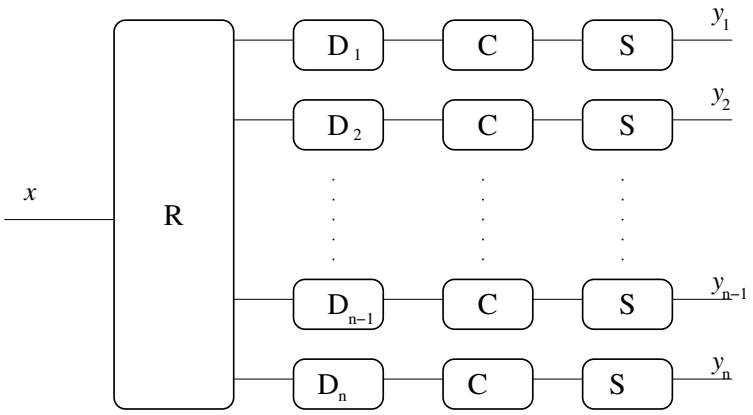


Figure 1. Direct model of a super-resolution problem.

where

- $\mathbf{x} \in \mathbb{R}^N$ is the original high resolution image
- for all $i = 1, \ldots, n$, $\mathbf{y}_i$ is a low resolution image
- $\mathbf{R}$ is a replication operator to create $n$ images from one original image
- for all $i$, $\mathbf{D}_i$ is a translation operator. That is if

$$\mathbf{g} = \mathbf{D}_i\mathbf{x}$$
$$g[p,q] = x[p - dx_i, q - dy_i] \ \forall p, q$$

  where $dx_i$ and $dy_i$ are the translations in the both direction (horizontal and vertical).
- $\mathbf{C}$ is the impulse response of the sensor (supposed to be known)

$$g = \mathbf{C}x$$
$$g[p,q] = \sum_n \sum_k c[p-n, q-n]x[p,q] \ \forall p, q$$

- $\mathbf{S}$ is a subsampling operator to produce a low resolution image from a high resolution image.

At end, we have for all low-resolution images $i$:

$$\mathbf{y}_i = \mathbf{SCD}_i\mathbf{x}$$

4.2. **From low to high resolution: an inpainting problem.** In a first time, we will suppose that the sensor is perfect, that is the convolution operator is simply the identity. The direct problem then reads for all $i$

$$\mathbf{y}_i = \mathbf{SD}_i\mathbf{x}$$

The goal is then to estimate the original high-resolution data $\mathbf{x}$ form its low resolution versions $\mathbf{y}_i$, that is several translated then subsampled version of $x$. For this, it is convenient to reformulate this problem as an inpainting problem. Indeed, one has:

$$\begin{aligned}\mathbf{S}^T\mathbf{y}_i &= \mathbf{S}^T\mathbf{S}\mathbf{D}_i\mathbf{x} + \mathbf{S}^T\mathbf{n}_i\\ \mathbf{z}_i &= \mathbf{M}_i\mathbf{x} + \mathbf{M}_i\mathbf{e}_i\end{aligned}$$

with $\mathbf{S}^T\mathbf{S}\mathbf{D}_i = \mathbf{M}_i$ is a binary mask of the size of the original high-resolution image, and $\mathbf{e}_i = \mathbf{S}^T\mathbf{n}_i$ the additive error due to noise. Now, in order to take into account all the data observed in all the $\mathbf{z}_i$, one can create one binary mask of the size of the original high-resolution image using the following procedure:

$$\begin{aligned}\mathbf{z} &= \frac{\sum_i(\mathbf{M}_i\mathbf{x} + \mathbf{e}_i)}{\sum_i \mathbf{M}_i}\\ &= \mathbf{M}(\mathbf{x} + \mathbf{e})\end{aligned}$$

where the division is element-wise, and only on non-zero values. $\mathbf{M}$ is then still a binary mask equivalent to $\bigvee_i \mathbf{M}_i$, and $\mathbf{e}$ is a noise but not necessarily white anymore. However, one can still consider than $\mathbf{e}$ is a white gaussian noise in practice.

## 5. Super-resolution: Inpainting + deconvolution

We are now able to write the super-resolution problem as a deconvolution problem followed by inpainting.

**Exercice 4.** *Write the equivalent direct problem, involving the deconvolution and the inpainting direct problems.*

5.1. **Given translations and down-sampling factor.** In a first time, we consider that we know the down-sampling factor, and the translations to get the low-resolution images. That is to say, the operator $\mathbf{M}$ is perfectly known.

**Exercice 5.**

*(1) Simulate the super-resolution problem with various input noise. From a high resolution image, fix a down-sampling factor and choose n horizontal translation and n vertical translation to generate n different low-resolution images.*
*(2) Create the equivalent binary mask* $\mathbf{M}$
*(3) Invert the problem with various approaches.*

5.2. **Translation estimation.** While on smart-phone, the position of the sensors are fixed and these translations can then supposed to be known, we consider here the general case where in practice, the translations are unknown. These translations can comes from small moves of the sensor or the object of consideration. For example, this technics is used by radar in order to improve the resolution of the license plate.

The translations are supposed to be very small: only few pixels. We can then apply an exhaustive technique of search of the offset maximizing the resemblance between the images. The most used criterion of resemblance is the correlation between the images. Indeed, this criterion is insensitive

to the variation of level in the image (insensitive to the difference of lighting of a scene). The correlation can be computed by:

$$\mathrm{Cor}(\mathbf{x}, \mathbf{y}) = \frac{\sum_{m,n}(x[m,n] - \bar{x})(x[m,n] - \bar{x})}{\sigma(x)\sigma(y)}$$

where

- $\bar{x}$ and $\bar{y}$ are the average value of the images $\mathbf{x}$ and $\mathbf{y}$
- $\sigma(x)$ and $\sigma(y)$ are the standard deviation of the images $\mathbf{x}$ and $\mathbf{y}$

In practice, one must fix the high-resolution factor and choose a reference image in order to estimate the translation with respect to this reference. The simplest choice is to use the first image. Then, for each images, we must estimate which translation $(dx, dy)$ gives the maximum correlation with the reference image. At end, we have an estimation of all the $D_i$. However, the translation $(dx, dy)$ must be lower than the value of 1 high-resolution pixel. In practice, one can create a bi-linear interpolation of the reference image in order to estimate the translations by correlation.

Once the translations are estimated for a given high-resolution factor, one can applies the same inversion techniques as in the previous subsection 5.1.

**Exercice 6.** *Re-use the previously simulated super-resolution problem. Try to invert-it by estimating the translation for various high-resolution factors.*

## 6. TO DO

**Provide:**

- A jupyter notebook or matlab publish which:
  - Simulate a super-resolution direct problem, given an impulse response, a sub-sampling factor and a noise level. It must provide a set of low resolution images given the high resolution image.
  - Invert any super-resolution direct problem, given an impulse response, a chosen upscaling factor and a estimated noise level.
- A short report of 4 pages double columns. You can use the IEEE Signal Processing conference template with a font size of 10, explaining the approach. You can follows intermediate questions of this document.

**Remark.** *Resolution of the proposed exercises is not mandatory. But it could really help to understand everything and providing a nice solution...*

**Evaluation:**

- Quality of the code
- Clarity of the script
- Clarity of the report
- Final performance of the inversion (SNR of the estimated image)
- Efficiency of the code